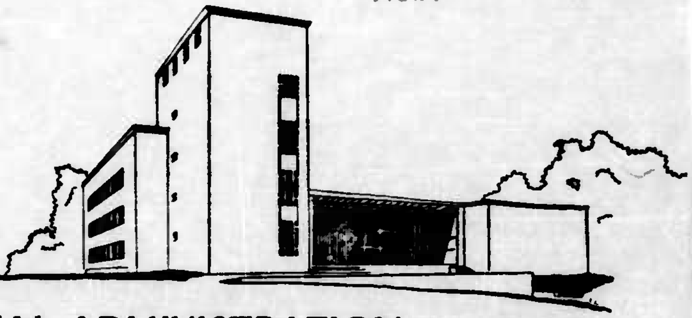
62-4-5

Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania



GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

O.N.R. Research Memorandum No. 86

MATHEMATICAL BASIS OF THE CRITICAL PATH METHOD*

by

F. K. Levy
G. L. Thompson
J. D. Wiest

May 30, 1962

Graduate School of Industrial Administration
Carnegie Institute of Technology
Pittsburgh 13, Pennsylvania

* The preparation of this paper was supported by the Office of Naval Research and the Bureau of Ships through grants to the Graduate School of Industrial Administration, Carnegie Institute of Technology.

~~1.~~ INTRODUCTION

The critical path method for planning projects, ~~introduced by J. E. Kelley, M. R. Walker, and others [3, 4]~~ has recently found wide-spread application in industry. The formal mathematical properties of the method can be approached from several points of view,[1, 2, 3, 5]. The purpose of this paper is to study it from the point of view of mathematical relations. It is interesting that the immediate precedence relation that is set up among jobs in a project should ideally be irreflexive and k-intransitive, in contrast to the properties usually possessed by mathematical relations (reflexivity and transitivity). This points up the need for more extensive study of the formal properties of mathematical relations by persons interested in management science. It is to this end that we have written the present paper.

In Section 2 we develop the properties of the relations "immediate predecessor" and "precedes" and the corresponding inverse relations, "immediate successor" and "succeeds." Next in Section 3 we discuss the critical path analysis of a project. Section 4 develops algebraic characterizations, in terms of matrices, of the relations studied earlier. Finally, in Section 5 we describe algorithms for checking for irreflexivity and k-intransitivity.

2. THE IMMEDIATE PREDECESSOR, IMMEDIATE SUCCESSOR, AND PROCEDES RELATIONS

The purpose of this section is to study the mathematical properties of the immediate predecessor, immediate successor, and precedes relations defined on the set of jobs in a project.

Let $J = \{a, b, c, \ldots\}$ be a set of jobs that must be done to complete a project. Let '<<' denote a relation between two jobs a and b in J, such that 'b << a' is defined for some pairs of jobs a and b, and is real "b is an immediate predecessor of a" or, equivalently, "a is an immediate successor of b". The interpretation of the statement "b << a" is that job b must be completed before job a can be started. Further, any given job can be started if and only if all its immediate predecessors have been completed. A project is the set J together with the relation <<.

Assumption 1. The immediate predecessor relation, <<, is irreflexive and k-intransitive for every k, that is,

(a) $a \ll a$ is false for all jobs a in J (irreflexivity).

(b) If $a = a_1 \ll a_2 \ll \ldots \ll a_k = b$, then it is false that $a \ll b$, for every pair of jobs a and b and every $k > 2$ (k-intransitivity).

Ordinary intransitivity corresponds to 3-intransitivity in the above definition. In Section 5 of this paper we shall develop tests for determining whether or not a given list of jobs satisfies Assumption 1.

DEFINITION. The set $P_a = \{b / b \ll a\}$ is the immediate predecessor set of job a. Similarly, the set $S_a = \{b / a \ll b\}$ is the immediate successor set of job a.

THEOREM 1. If Assumption 1 holds, then the set P_a is the smallest set of jobs in J that must be completed before a can be started. Similarly, S_a is the smallest subset of jobs in J that cannot be started until job a is completed.

PROOF. Since << is irreflexive, a is not an element of P_a. Hence when all jobs in P_a have been completed, job a can be started.

Now suppose, contrary to the assertion, that job a can be started before some job x in P_a is known to be completed. Since $x \ll a$, this can happen only if the completion of job x is implied by the completion of some other job b in P_a. That is, there must be a relation of the form

$$x = b_1 \ll b_2 \ll \dots \ll b_k = b \ll a$$

for some b in P_a. But since $x \ll a$, we have a contradiction of the assumption of k-intransitivity of $\ll$. This completes the proof of the first assertion. The proof of the other assertion is similar.

The <u>project graph</u> G of a project is a planar graph with points in the plane representing jobs, and a directed line segment connecting two jobs, a and b, if and only if $a \ll b$ holds. For examples of project graphs see [3, 5]. A <u>path</u> in G is a set of nodes $a_1, a_2, \dots, a_n$ for which the immediate predecessor relation holds as follows:

$$a_1 \ll a_2 \ll \dots \ll a_n.$$

A cycle in G is a closed path of the form

$$a_1 \ll a_2 \ll \dots a_{n-1} \ll a_n = a_1.$$

A project graph G is <u>acyclic</u> if and only if it has no cycles.

From the relation $\ll$ we can derive another relation $<$ as stated in the next definition.

DEFINITION. $a < b$, read "a precedes b" (or equivalently "b succeeds a") if and only if there is a set of jobs $\{c_1, c_2, \dots, c_n\}$ where $n > 2$ such that

$$a = c_1 \ll c_2 \ll \dots \ll c_n = b.$$

In other words, a precedes b, if and only if there is a path from a to b in the project graph G. Note that k-intransitivity can be stated as: if $a < b$ then it is false that $a \ll b$.

ASSUMPTION 2. The precedes relation $<$ is asymmetric, that is, if $a < b$ then it is false that $b < a$, for all a and b in J.

REMARK. Obviously if G is acyclic, then $<$ is also irreflexive.

In Section 5 we shall develop effective tests for seeing whether a given set of jobs and an immediate predecessor relation define an asymmetric precedes relation.

DEFINITION. A relation that is transitive and asymmetric is said to be a <u>preference relation</u>.

THEOREM 2. If Assumption 2 holds, then the precedes relation $<$ is a preference relation, and the graph G is acyclic.

PROOF. We prove first that $<$ is transitive. Suppose $a < b$ and $b < c$. Then there are paths

$$a = d_1 \ll d_2 \ll \dots \ll d_k = b$$

$$b = e_1 \ll e_2 \ll \dots \ll e_h = c.$$

But then it is clear that

$$a = d_1 \ll d_2 \ll \dots \ll d_k = b = e_1 \ll e_2 \ll \dots \ll e_h = c$$

is a path from a to c so that $a < c$.

Since $<$ is asymmetric (by assumption) and transitive (as just proved) then it is a preference relation. On the other hand, if "precedes" is a preference relation then, by definition, it is asymmetric.

Suppose that the project graph G contained a cycle of the form

$$a = b_1 \ll b_2 \ll \dots \ll b_n = a,$$

where $n > 2$. Hence if we let $b = b_j$ where j is between 1 and n, we see that $a < b$ and $b < a$, which contradicts the asymmetry of the relation $<$. Therefore G is acyclic.

LEMMA 1. The graph G of a project is cyclic if and only if there is a subset Q of J such that $P_a \cap Q \neq \emptyset$ for all a in Q.

PROOF. If G contains a cycle, let Q be the set of jobs that make up the cycle. Then it is obvious that Q satisfies the requirement of the lemma.

Conversely, let Q be a subset of J with the property stated. We will show that G contains a cycle. Let a_1 be any element of Q; let a_2 be any element in $P_{a_1} \cap Q$; let a_3 be any element of $P_{a_2} \cap Q$, etc. Since J and hence Q have only a finite number of elements, we must eventually choose an element that has already been chosen before. We then have

$$a_1 << a_2 << \dots << a_n$$

together with the fact that $a_n = a_j$ for some $j < n$. Hence G has a cycle.

DEFINITION. An ordered <u>job list</u> $J^* = [a_1, a_2, \dots, a_m]$ is obtained from a set of m jobs $J = \{a, b, \dots\}$ by listing them in a definite order. An ordered job list J^* will be said to be in <u>technological order</u> if no job appears on the list until all of its predecessors have already appeared on the list.

THEOREM 3. Assumption 2 holds if and only if it is possible to list the jobs in J in a technologically ordered job list J^*.

PROOF. Let S_0 be the set of jobs in J that have no immediate predecessors. S_0 must be nonempty or else at least one subset of J would have the property of Lemma 1 and G would by cyclic, contradicting Theorem 2. Let S_1 be the set of jobs in J whose immediate predecessors are all in S_0; let S_2 be the set of jobs in J whose immediate predecessors are in $S_0 \cup S_1$; etc.; in general, let S_k be the subset of jobs in J all of whose predecessors are in $S_0 \cup S_1 \cup \dots \cup S_{k-1}$. Since J is a finite set this process will stop, with say $k = n$.

Suppose now that there is a nonempty subset Q of J of elements that are not in any of the sets $S_0, \ldots, S_n$ so constructed. We shall show that Assumption 2 is violated. Every element a in Q must have the property that $P_a \cap Q \neq \emptyset$, since otherwise job a would have been a member of one of the sets S_k But then Q has the property of Lemma 1, and G is cyclic. Since Theorem 2 says that Assumption 2 holds if and only if G is acyclic, it follows that Q must be empty and

$$J = S_0 \cup S_1 \cup \ldots \cup S_n .$$

We can now construct a technologically ordered listing J^* of J by first listing all the elements of S_0 in any order, then all the elements of S_1 in any order, etc., until we have listed all the elements in S_n, hence all the elements in J.

If, on the other hand, J can be listed as a technologically ordered job list J^*, then defining the sets S_0, S_1, ..., S_n, and Q as before, it is obvious that Q is empty, so that Assumption 2 holds.

3. JOB TIMES: CRITICAL JOBS: CRITICAL PATHS

Jobs take a certain amount of time to complete. Let t_a be the time to complete job a. For purposes of this article we assume that job times are known with certainty.

It is convenient (though not necessary) to introduce two fictitious jobs, Start and Finish, both of which have zero job times. The job, Start, is defined to be the unique predecessor of all jobs in J that do not have predecessors in J, and the job, Finish, is defined to be the unique successor of all jobs in J that do not have successors in J. These two jobs have the property of closing up the ends of the project graph. Start and Finish are added to the set J.

Suppose now that the project proceeds and every job in the project is started as soon as all of its immediate predecessors are finished. It is then possible to compute an early start (ES) for each job in the project, and also an early finish (EF) time for each job. We describe an algorithm that will calculate ES(a) and EF(a) for each job a in J.

ALGORITHM C_1.

1. Define ES and EF of Start both to be zero.*/
2. Let a be any job such that the early finish EF times of all jobs in P_a have already been computed.

   Then compute

   $$ES(a) = \underset{s \text{ in } P_a}{\text{Max}} \quad EF(x)$$

   and also

   $$EF(a) = ES(a) + t_a.$$
3. Eventually the early finish time, F, of Finish will be computed.

*/ Alternatively, any arbitrary start time (say S) can be used rather than 0 for the ES and EF of Start.

THEOREM 4. If Assumption A_2 holds, then

(a) Algorithm C_1 will assign unique early start and early finish times to all jobs in J.

(b) For any job a in J the early finish time EF(a) is the sum of the job times on the longest path from Start to and including a.

(c) The earliest time that all jobs in the project can be finished is F.

PROOF. Since Assumption A_2 holds, the jobs can be put in a technologically ordered list. We prove the first two parts of the theorem by induction on the kth element, a_k, of the list.

Proof of (a). The first job in the list is Start which is always assigned early start and early finish times of 0. Suppose now that (a) is true for

$a_1 \ldots, a_{k-1}$. We want to show it true for a_k. Since the predecessors of a_k are already on the list and have unique early finish times, their maximum which is the early start time assigned by the algorithm to a_k, is also unique. But then so is the early finish time of a_k, since it is obtained by adding the job time of a to the early start time of a.

Proof of (b). The statement is obviously true for the first job, Start. Suppose it is true for the first k-1 jobs. Then, since the early start time of a_k is the maximum of the early finish times of its predecessors, there is a path whose length is equal to the early start time of a from Start up to a. Then since it takes time t_a to complete a, it follows that there is a path of length EF(A) from Start to the end of a, and this is the longest such path.

Proof of (c). Since Finish is a job on the list, statement (b) is true for Finish. Hence Finish can be completed at time F. Since Finish has no successors, the entire project is also complete.

Projects usually have due dates or target dates T, by which they must be completed. The only achievable target dates satisfy $T \geq F$, since F is the earliest time at which all the jobs in the project can be completed. If we know a target date T, then working backwards from the end of the project, we can compute the latest time at which each job in the project can be completed in order not to delay the entire project beyond the time T. We call this the latest finish (LF) time of the job. From this we can also deduce a late start (LS) time for each job. We next describe an algorithm to compute these times.

ALGORITHM C_2.

1. Define the LF and LS of Finish both to be T.
2. Let a be any job such that the LS times of all jobs in S_a have already been computed. Then compute

$$LF(a) = \min_{x \text{ in } S_a} LS(x)$$

and also

$$LS(a) = LF(a) - t_a.$$

3. Eventually the late start time L of Start will be computed. The quantity L will be shown to be the slack time of the project. (If S is not 0, then slack time of the project is L - S).

THEOREM 5. If Assumption A_2 holds then Algorithm C_2 will assign unique late start and late finish times to all jobs in J.

PROOF. The details are omitted, but the proof is similar to that for theorem 4(a).

DEFINITION. For any job a, the quantity

$$SL(a) = LS(a) - ES(a) = LF(a) - EF(a)$$

is defined to be the slack time (or simply slack) of job a.

Intuitively, the slack of a job a is the maximum amount that the completion of a can be delayed without necessarily delaying the completion of the entire project.

DEFINITION. By a critical job we shall mean a job having minimum slack time.

Obviously every project has at least one critical job.

LEMMA 2. If assumption A_2 holds, then

(a) Every critical job in J except Start and Finish has at least one critical job as a predecessor and at least one critical job as a successor.

(b) Finish and Start are critical jobs.

(c) $L = T - F$ is the minimum slack time, i.e., the slack time of all critical jobs.

PROOF. (a) Suppose, on the contrary, that there is a critical job a (other than Start) such that none of its predecessors x is critical. Then, since a is a successor of x, it is true that

$$(1) \qquad LS(a) \geq LF(x).$$

Also, since a is critical while x is not, we have

$$(2) \qquad LS(a) - ES(a) < LF(x) - EF(x).$$

Combining (1) and (2) we have

$$LF(x) - ES(a) < LF(x) - EF(x),$$

which implies,

$$(3) \qquad ES(a) > EF(x)$$

for all predecessors x of a. However, since $ES(a)$ is defined to be the maximum of all the early finish times of its predecessors, we see that (3) contradicts this definition. Hence at least one predecessor of a is critical. The proof that at least one successor of a is critical follows similarly.

(b) Let a be any critical job in J. It has at least one critical successor; pick one, say it is a_1. Now pick a critical successor of a_1, say it is a_2, etc. Eventually, Finish must be chosen as the critical successor of some job so that Finish is critical. Similarly, Start is critical.

(c) Since Finish and Start are both critical it follows that $L = T - F$ is the minimum slack time of all jobs in the project.

THEOREM 6. If Assumption A_2 holds, then

(a) There is at least one path, called a <u>critical path.</u> From Start to Finish such that every job on the path is critical.

(b) Every critical job lies on such a path.

(c) The sum of the job times on every critical path is F which is at least as large as the sum of the job times on every other path from Start to Finish.

PROOF. Let a be any critical job in J. Then, as in the proof of

Theorem 5(b), pick a critical successor a_1 and a critical predecessor b_1 of a; then pick a critical successor a_2 of a_1 and a critical predecessor b_2 of b_2; etc. Continue until Finish and Start are chosen. The result is a critical path from Start through a to Finish. This proves the first wo statements. Statement (c) is simply a restatement of Theorem 4(b) for the job Finish.

4. PREDECESSOR AND SUCCESSOR MATRICES

In this section we shall develop some algebraic methods for studying the predecessor and precedes relations. We shall give algebraic characterizations of Assumptions 1 and 2 of Section 2.

In Section 2 we defined a path in the project graph G to be a set of k jobs satisfying $a_1 \ll a_2 \ldots \ll a_k$. Two paths are different if they differ in the identity or order of jobs on the path. We shall say that the <u>length</u> of such a path is k.

Suppose now that our project consists of $\overset{\text{an ordered}}{\wedge}$ set $J=[a_1, a_2, \ldots a_n]$ of n jobs, including Start, a_1, and Finish, a_n. We define a predecessor and a successor matrix for the project.

DEFINITION. The <u>predecessor matrix</u>/$\overset{P}{}$for the project is the nxn matrix with components p_{ij} defined as follows:

$$p_{ij} = \begin{cases} 1 & \text{if } a_i \ll a_j \\ 0 & \text{otherwise.} \end{cases}$$

The <u>successor matrix</u> S for the project is defined to be the transpose of the matrix P.

Since $s_{ij} = p_{ji}$ it follows that

$$s_{ij} = \begin{cases} 1 & \text{if } a_j \ll a_i \\ 0 & \text{otherwise.} \end{cases}$$

By definition S and P are matrices having only 0 and 1 as entries. We shall concentrate our study on the matrix P, and at the end of the section make some remarks about the matrix S.

Since P is a square matrix we can take powers of it, P^n. We denote the i,jth entry of P^n by $p_{ij}^{(n)}$.

THEOREM 7. If $p_{ij}^{(k)} = m > 0$, then there exist m distinct paths of length k from a_i to a_j.

PROOF. We prove the theorem by induction on k. The statement is true for k = 0 and 1. Suppose it is true for the entries of P^{k-1}; we want to show it is true for the entries of P^k. Since $P^k = P^{k-1}.P$ we see that

$$(4) \qquad p_{ij}^{(k)} = p_{i1}^{(k-1)}p_{1j} + p_{i2}^{(k-1)}p_{2j} + \ldots + p_{in}^{(k-1)}p_{nj}.$$

Since all of the terms in this sum are nonnegative, it follows that $p_{ij}^{(k)}$ is positive only if one or more of the terms $p_{ih}^{(k-1)}p_{hj}$ is positive. And the latter can happen only if $p_{ih}^{(k-1)} > 0$ and $p_{hj} > 0$. By the induction hypothesis $p_{ih}^{(k-1)}$ gives the number of distinct paths of length k-1 from a_i to a_h, and since $p_{hj} = 1$, we can construct the same number of distinct paths of length k from a_i to a_j. Moreover, if $p_{ih}^{(k-1)} > 0$ and $p_{im}^{(k-1)} > 0$, where $h \neq m$, the paths so constructed are distinct. Hence the sum in (4) counts exactly the number of distinct paths from ~~ℓ~~ a_i to a_j.

COROLLARY. If there is a longest path i.e., a path containing the most nodes in the project graph G, and it is of length N, then $P^N \neq 0$ and $P^{N+1}=0$.

Theorem 7 holds regardless of whether or not Assumptions 1 and 2 hold. In the remaining theorems we investigate the consequences of these assumptions.

THEOREM 8. Assumption 1 can be characterized as follows:

(a) The precedes relation $\ll$ is asymmetric if and only if $p_{ij} = 1$ implies $p_{ji} = 0$, or equivalently, the maximum entry in P + S is 1.

(b) The precedes relation $\ll$ is k-intransitive for every ~~ℓ~~ k if and only if $p_{ij} = 1$ implies $p_{ij}^{(k)} = 0$ for all $k > 2$.

PROOF. (a) If $<<$ is asymmetric then $a_i << a_j$ implies that it is false that $a_j << a_i$ hence $p_{ij} = 1$ and $p_{i} = 0$, and conversely.

(b) Since $p_{ij} = 1$ implies that there is a path of length 1 from a_i to a_j and $p_{ij}^{(k)}>0$ implies that there is at least one path of length k from a_i to a_j, it follows that k-intransitivity implies that not both kinds of paths are permissible. The converse is also true.

The results of this theorem show that by examining P and its powers we can find out whether or not there are redundant predecessors in the project list.

THEOREM 9. Assumption 2 holds if and only if $P^{N+1} = 0$ for some N.

PROOF. If Assumption 2 holds then there are no cycles in the project graph. Hence all paths must have finite length (since the number of jobs is finite) and therefore, by the corollary to Theorem 7 $P^{N+1} = 0$ for some integer N. Conversely, if $P^{N+1} = 0$ for some N, then there is no path longer than N, hence there cannot be any cycles and Assumption 2 holds.

DEFINITION. If P is a square matrix then the <u>Neumann inverse</u> of the matrix $I - P$ is defined to be

(5) $$(I - P)^{-1} = I + P + P^2 + \dots + P^n + \dots,$$

when the series on the right converges.

LEMMA 3. Let P be a matrix with integer entries; then the Neumann inverse of $I - P$ exists if and only if $P^{N+1} = 0$ for some $N > 0$.

PROOF. The infinite series in (5) can converge only if $P^n \dashrightarrow 0$ with increasing n. However, since P and P^n have integer entries, this can happen only if $P^{N+1} = 0$ for some N, which proves the necessity of the condition.

To prove sufficiency, consider the following identity

(6) $$(I - P)(I + P + P^2 + \dots + P^n) = I - P^{n+1},$$

which can be established by induction. From (6) it is clear that if $P^{N+1} = O$, then $(I - P)^{-1}$ exists and

$$(I - P)^{-1} = I + P + P^2 + \dots + P^N.$$

This completes the proof of the theorem.

THEOREM 10. Assumption 2 holds if and only if the Neumann inverse Q exists and equals

(7) $$Q = (I - P)^{-1} = I + P + P^2 + \dots + P^N,$$

that is, if and only if the project graph G has a longest path.

PROOF. These statements are consequences of Theorems 7, 9, and Lemma 3.

The entries of the matrix $Q = I + P + P^2 + \dots + P^N$ have interesting interpretations. Clearly, q_{ij} is equal to the total number of paths, of all lengths, from a_i to a_j. This enables us to give still another characterization of Assumption 1.

THEOREM 11. Assumption 1 holds if and only if $p_{ij} = 1$ implies $q_{ij} = 1$.

PROOF. If both $p_{ij} = 1$ and $q_{ij} = 1$, then there is exactly one path, which is necessarily of length 1, from a_i to a_j. It follows that $<<$ is both irreflexive and k-intransitive. The proof of the converse is similar.

There is a close connection between the results of this section and those of B. Giffler in his schedule algebra paper (2). For if we put p_{ij} equal to the time of a_i when $a_i << a_j$, and redefine the matrix operations as Giffler does, then the entries of Q will be the maximum path from a_i to a_j. This technique can be used to make the critical path calculations of Section 3.

5. REDUNDANT PREDECESSOR AND CYCLE CHECKS

The results of the preceding section can be immediately translated into algebraic checks for redundant predecessors and cycles in the project graph G. We state these as an algorithm. Assume that we are given a job list $J = \{a, b, \ldots\}$ and for each job a set of immediate predecessors P_a. We define an algorithm R_1 for cycle and redundant predecessor checking:

ALGORITHM R_1.

(i) Set up the precedence matrix P.

(ii) Calculate $Q = (I - P)^{-1}$ if it exists. If it does not exist then G has a cycle and the original job data should be checked. (The Neumann inverse $(I - P)^{-1}$ may be computed by one of the standard matrix inversion routines, or else by various iterative schemes that quickly compute the partial sums $I + P + \ldots + P^k$.)

(iii) If for some i and j both $p_{ij} = 1$ and $q_{ij} > 1$, then job a_j has redundant predecessors and the original job data should be checked. Computation of the powers P^k are useful in making this check.

Still another set of algorithms for checking for cycles and redundant predecessors can be derived from the results of Section 2. These algorithms are probably less wasteful of computer memory space than the matrix algorithms just described. We list the steps of the algorithm (R_2) below.

ALGORITHM R_2.

1. List the jobs in any order.

2. Pick the set P_0 of jobs with no predecessors (except Start).
3. For i going from 1 to N pick the set P_i of jobs all of whose predecessors are in P_{i-1}. Here N is the largest integer such that P_N is not empty.
4. If there are jobs in J that are not included in any of the sets $P_0, P_1, \ldots, P_N$, then the original job data contains a cycle and should be checked. Otherwise go to 5.
5. For k going from 2 to N, consider each job a in P_k. If a has a predecessor b in P_{k-1}, include in the predecessors of a all the predecessors of b. (At the conclusion of this step, each job will have a complete list of its predecessors, and hence there will be maximum redundancy in the predecessor lists. This redundancy is removed in step 6.)
6. For h going from 1 to N, consider a job a in P_h. If a is a predecessor of some job b in P_k (where $k > h$), then remove from P_b any prerequisites of a, i.e., replace P_b by $P_b - P_a$.
7. At the end of 6, all jobs will have only immediate predecessors listed.

The proofs that Algorithm R_2 will check for cycles and will remove redundant predecessors are based on the material of Section 2.

BIBLIOGRAPHY

[1] Charnes, A., and W. W. Cooper, "A Network Interpretation and a Directed Sub-dual Algorithm for Critical Path Scheduling," forthcoming in the Journal of Industrial Engineering.

[2] Giffler, B., "Schedule Algebras and Their Use in Formulating General Systems Simulations," forthcoming in Factory Scheduling, by J. F. Muth and G. L. Thompson (Eds.) Prentice-Hall, Inc., 1963.

[3] Kelley, J. E., Jr., "Critical-Path Planning and Scheduling: Mathematical Basis," Operations Research, 9, (1961) 296-320.

[4] ------------------ and M. R. Walker, "Critical-Path Planning and Scheduling: an Introduction," Proc. Eastern Joint Computer Conference, (1959) 160-173.

[5] Levy, F. K., G. L. Thompson, and J. D. Wiest, "Critical Path Method -- A New Tool for Management," forthcoming.

[6] Levy, F. K., G. L. Thompson, and J. D. Wiest, "Multi-Ship, Multi-Shop Workload Smoothing Program," Naval Research Logistics Quarterly, 8 (1962).